Scheduling in the context of Automatic Hardware Generation

Razvan Nane*, Koen Bertels *

* *Computer Engineering, Postbus 5031, 2600 GA, Delft, The Netherlands*

ABSTRACT

The advantages of reconfigurable technology in terms of performance have been widely recognized. Due to its potential to greatly accelerate a wide variety of applications, reconfigurable computing has become a subject of a great deal of research. Its key feature is the ability to perform computations in hardware to increase performance, while retaining much of the flexibility of a software solution. In this context, the Delft Workbench group focuses on software and back end compilers for Reconfigurable Embedded Processors. Its objective is a semi-automatic tool platform for integrated hardware-software co-design targeting heterogeneous computing systems containing reconfigurable components. Delft Workbench addresses the entire design cycle rather than isolated parts. It involves the development of compilers for reconfigurable platforms, programming models, hardware software co-design, CAD and design space exploration software, optimization algorithms and integration software development.

The current research emphasis falls onto four major categories: Code Profiling and Cost Modeling in which the segments of the input application that are suitable for hardware execution are identified and preliminary estimation of the implementation costs is performed. Graph Transformations where the high-level application is restructured in function of the reconfigurable platform through possible algebraic transformations. Retargetable compilation: Specific compiler optimizations are designed to meet the specific requirements of the reconfigurable components; and VHDL generation where for preliminary costs estimation and hardware implementation of non-critical parts, the possibility of automated VHDL generation is investigated.

Within this context, the DWARV (C-to-VHDL) tool was designed. The motivation for it was to have something that would allow us to estimate performance and prototype fast, and to allow application designers without hardware knowledge to choose between software or hardware implementations and not be restricted only to software solutions.

The toolset provides support for broad range of application domains and exploits the operation parallelism, available in the algorithms. Having no limitations on the application domain and being able to actually execute on a real hardware prototyping platform were the most important design goals of DWARV and the things that basically makes this tool different than similar projects like SystemC, ImpulseC and others. The input of the toolset is pragma annotated C code without any syntax extensions. A number of restrictions on the C-language currently apply but will be relaxed in the future. However, those restrictions, do not limit the application domains. In its current state, with only a limited number of available optimizations, DWARV is capable

[1]E-mail: rnane@ce.et.tudelft.nl,k.l.m.bertels@tudelft.nl

of providing the following functionality: No limitations of the application domains. Algorithms with different characteristics are automatically translated and executed; Kernel-wise speedups of 9.7 times over the software execution; Substantial overall application speedup of up to 6 times over software execution is observed; High performance efficiency. The achieved speedup amounts from 13% to 94% of the theoretically possible maximum speedup, constituted by Amdahl's law; Actual execution on a real hardware prototype platform.

Scheduling is an important step in this respect in high-level synthesis and means assigning each operation in the input description to a time step in such a way that given constraints and optimization objectives are satisfied, i.e. minimize execution time while not exceeding pre-determined resource constraints. In the context of DWARV project, this implies: Maximize parallelism, Balance operations load, Balance storage, Reduce interconnect, Reduce control. Scheduling is therefore an aspect of utmost importance and is the current topic of research in DWARV. Before describing the problem, some project terminology and notations needs to be referred to.

The output of the first step in DWARV, i.e. Graph building, is a Hierarchical Data-Flow Graph (HDFG) containing different types of nodes and edges. Nodes can be of the following types: data processing (DPN), data transfer (DTN) or data storage (DSN); whereas edges can be data-explicit(EDDE) or -implicit(IDDE) and control dependencies(CDE). DPN can be atomic (e.g. Addition) or compound (e.g. function, loop with unknown iterations) therefore the execution delay of these types of nodes can be known or unknown. Disregarding for the time being the other types of nodes and the edges, we observed that the compound nodes are at the base of the scheduling problem.

In order to be able to efficiently schedule the graph, we need to reduce its complexity (decompose it) by finding a cut that will give optimal or near optimal results. Nonetheless, this is not achievable in the current situation, as we are dealing with non-deterministic nodes with unknown delays. Hence the difficulty of imposing some constraints on the graph, which would lead to a cut that would generate better results than those obtained by running the non-deterministic nodes sequentially. With other words, is there a way to schedule in parallel non-deterministic and deterministic nodes in a graph and if so, what are the implied conditions. One possible solution that is currently being investigated is to try to find some statistics for the non-deterministic nodes based on profiling and to schedule accordingly in order to investigate the performances of such solution.

KEYWORDS: ACES; poster session; reconfigurable; scheduling; hardware generation